Spring4D

european conference

Interception and dynamic proxy

About me

Primož Gabrijelčič

http://primoz.gabrijelcic.org

- programmer, MVP, writer, blogger, consultant, speaker

Blog	*http://thedelphigeek.com*
Twitter	*@thedelphigeek*
Skype	*gabr42*
LinkedIn	*gabr42*
GitHub	*gabr42*

european 4D conference

Today’s topics

- Mocking
  - Unit testing
  - Prototyping
- Dynamic proxy
  - Delphi as a DSL

Mocking

Mocking

- Dynamic object implementing an interface
- Pluggable functionality
- Assertion testing (unit tests)

Mock<I>

- Creation phase
  - .Create
- Setup phase
  - .Setup
- Execution phase
  - Call method of *I*
- Assertion phase
  - .Received

Creation phase

```
uses    Spring.Mocking

type    Intf = interface …

var mock: Mock<intf> := Mock<Intf>.Create;

mock := Mock<Intf>.Create(TMockBehavior.Strict);
```

european 4D conference

Setup phase

```
.Setup
    .Executes([action])    ⎫
    .Raises<TException>    ⎬ .When([condition]).<Intf method>
    .Returns<T>            ⎭
```

Sequences

```
var sequence: MockSequence;
.Create(TMockBehavior.Strict)
.Setup(MockSequence)
.Returns([value1, value2, … valueN])

sequence.Completed
```

Parameter matching

Arg

.IsAny<T>, .IsEqual<T>, .IsInRange<T>, .IsIn<T>,
.IsNotIn<T>, .IsNil<T>, .IsNotNil<T>,
.IsRegex

Specification

- Multiple rules for the same mocked method
- Generic first, specific last

Assertion phase

.Received([times], [match]).<intf method>

Times

.Any, .AtLeast, .AtLeastOnce, .AtMost, .AtMostOnce,

.Between, .Exactly, .Never, .Once

Dynamic proxy

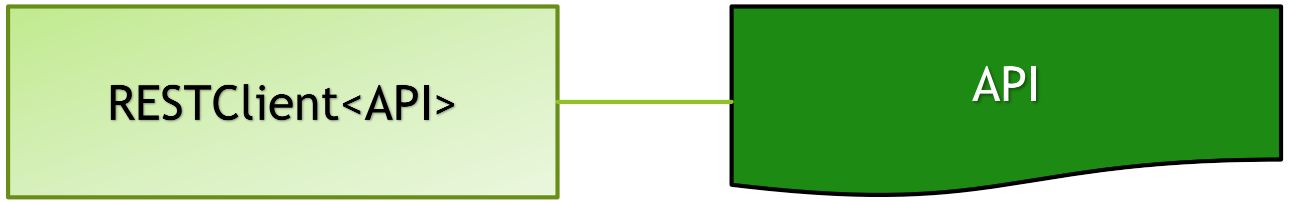
“

THE REST CLIENT STORY

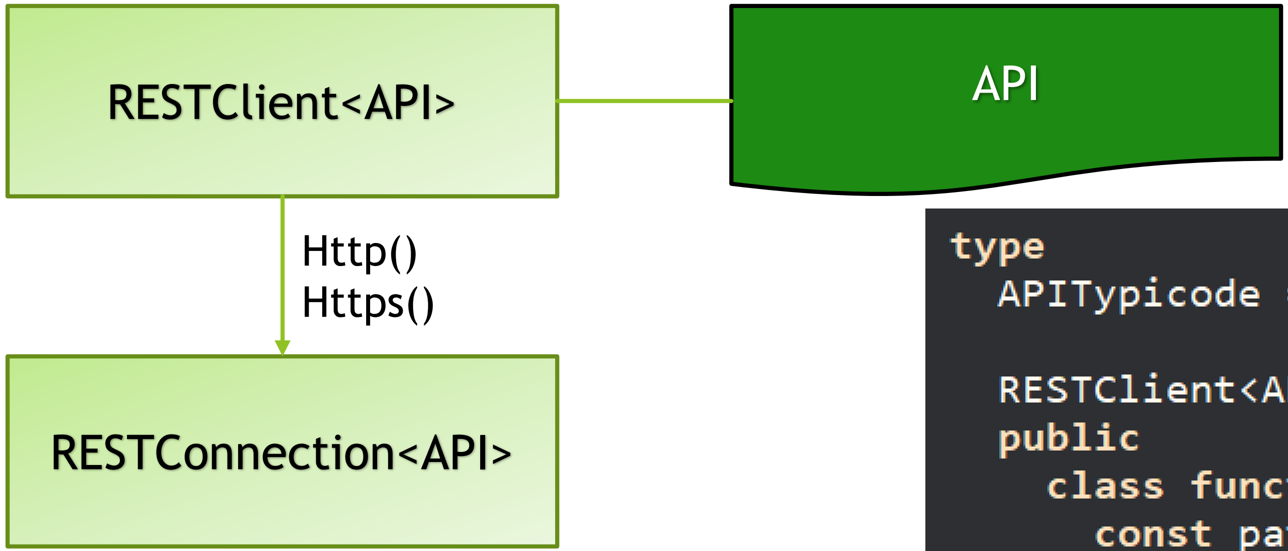
”

API

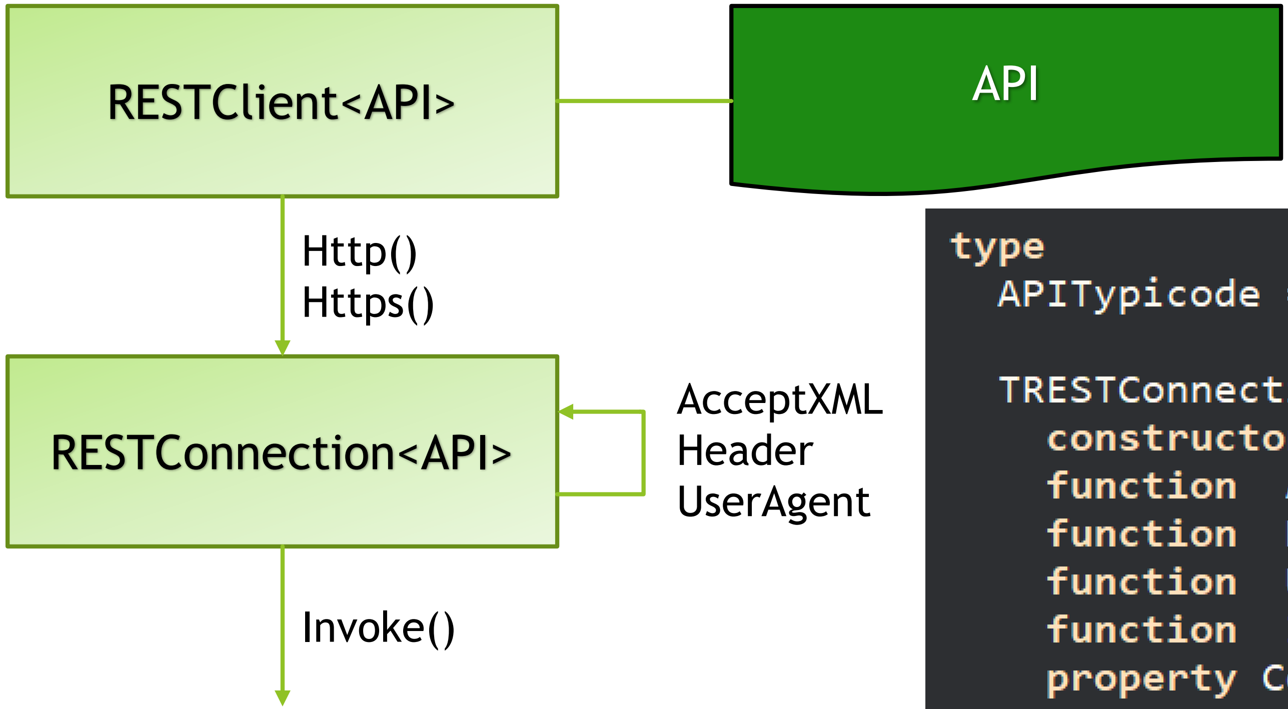
```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-
    function  Albums: IHTTPResponse; overload;
    function  Albums(id: integer): IHTTPResponse; overload;
    function  Comments: IHTTPResponse; overload;
    function  Comments(id: integer): IHTTPResponse; overloac
  end;
```



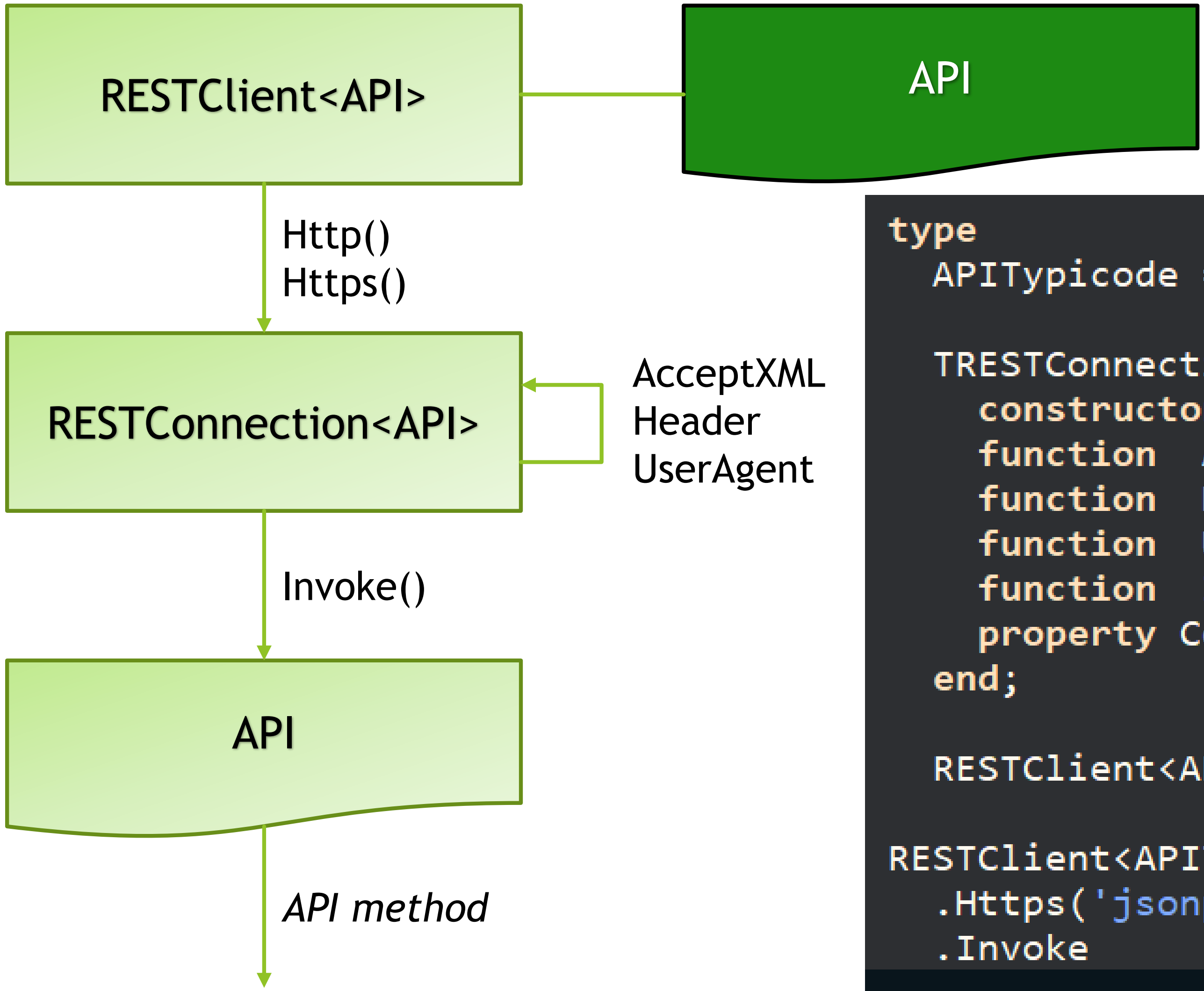
```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-
    function  Albums: IHTTPResponse; overload;
    function  Albums(id: integer): IHTTPResponse; overload;
    function  Comments: IHTTPResponse; overload;
    function  Comments(id: integer): IHTTPResponse; overloac
  end;

RESTClient<APITypicode>
```

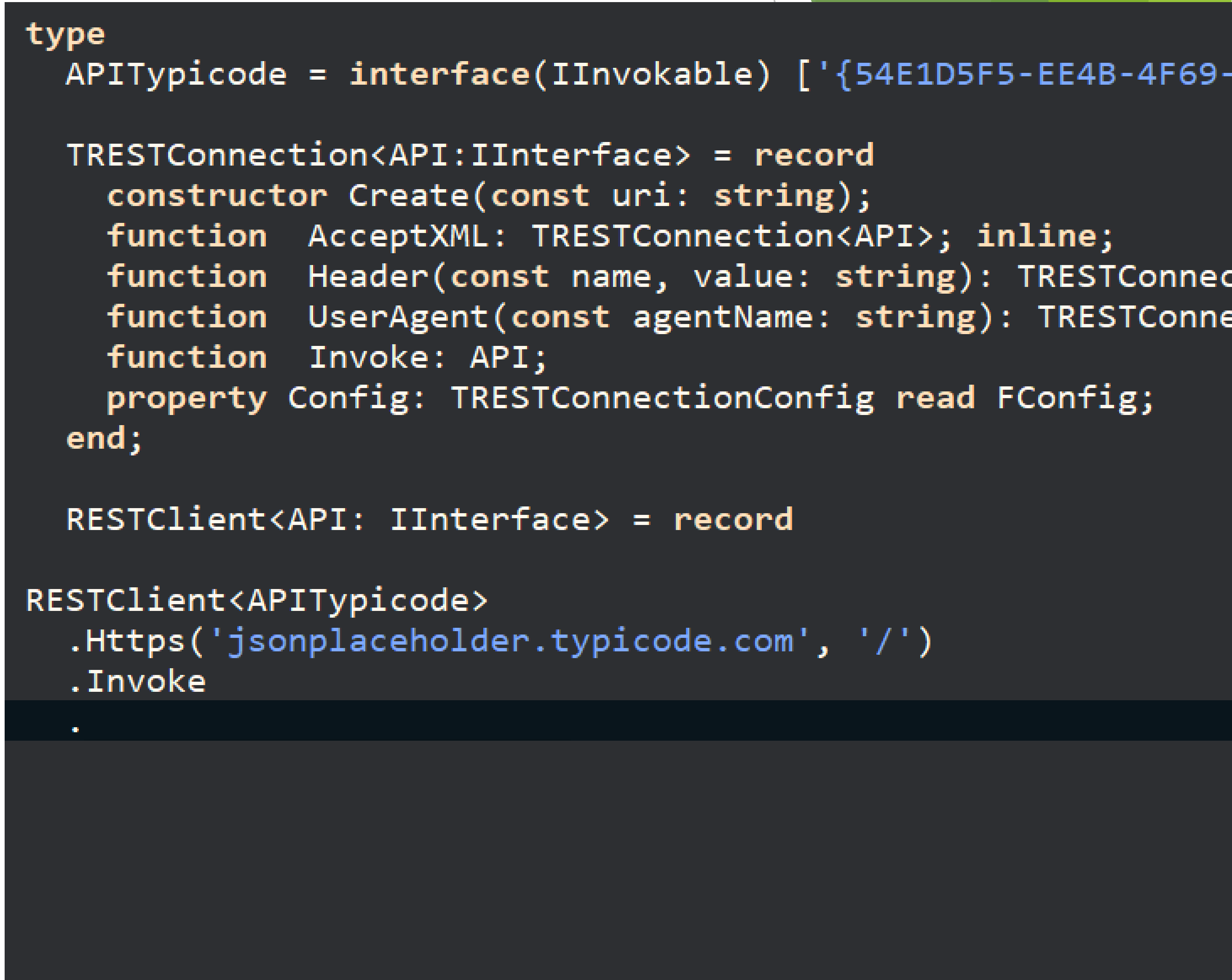


```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-

  RESTClient<API: IInterface> = record
  public
    class function Https(const host: string; port: integer;
      const path: string):
      TRESTConnection<API>; static;
  end;

RESTClient<APITypicode>
```

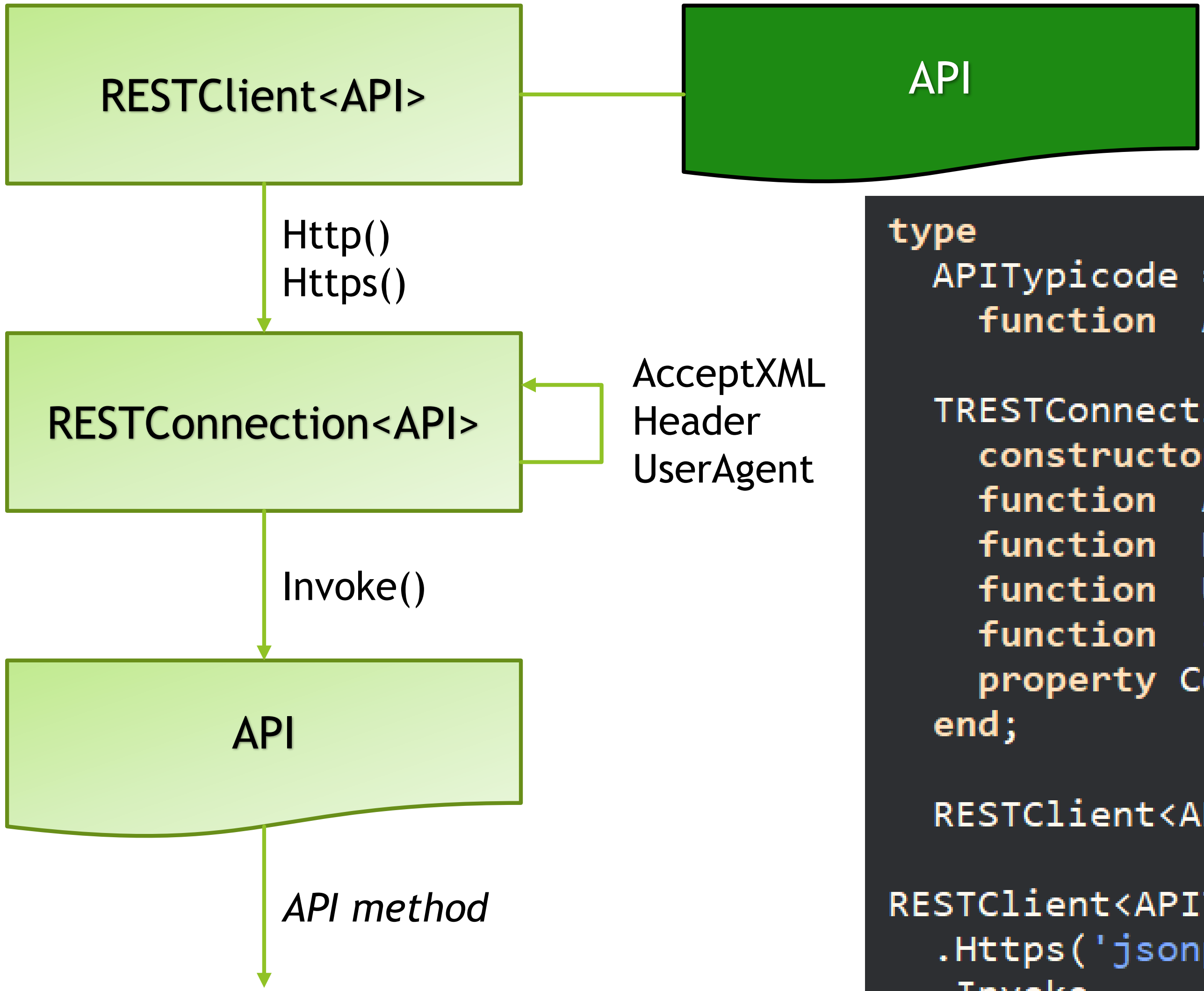


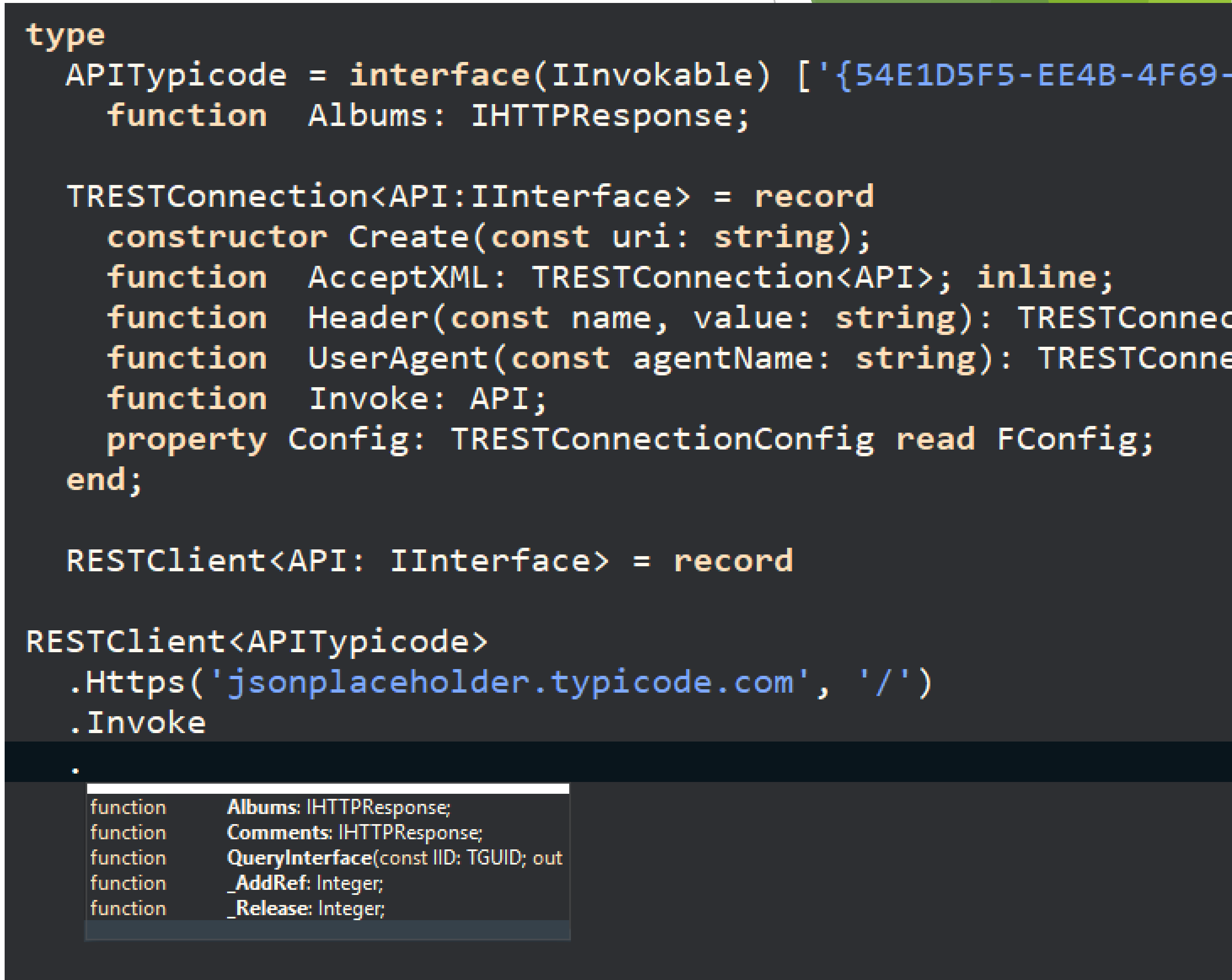
```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-

  TRESTConnection<API:IInterface> = record
    constructor Create(const uri: string);
    function  AcceptXML: TRESTConnection<API>; inline;
    function  Header(const name, value: string): TRESTConnec
    function  UserAgent(const agentName: string): TRESTConne
    function  Invoke: API;
    property Config: TRESTConnectionConfig read FConfig;
  end;

  RESTClient<API: IInterface> = record

RESTClient<APITypicode>
  .Https('jsonplaceholder.typicode.com', '/')
```





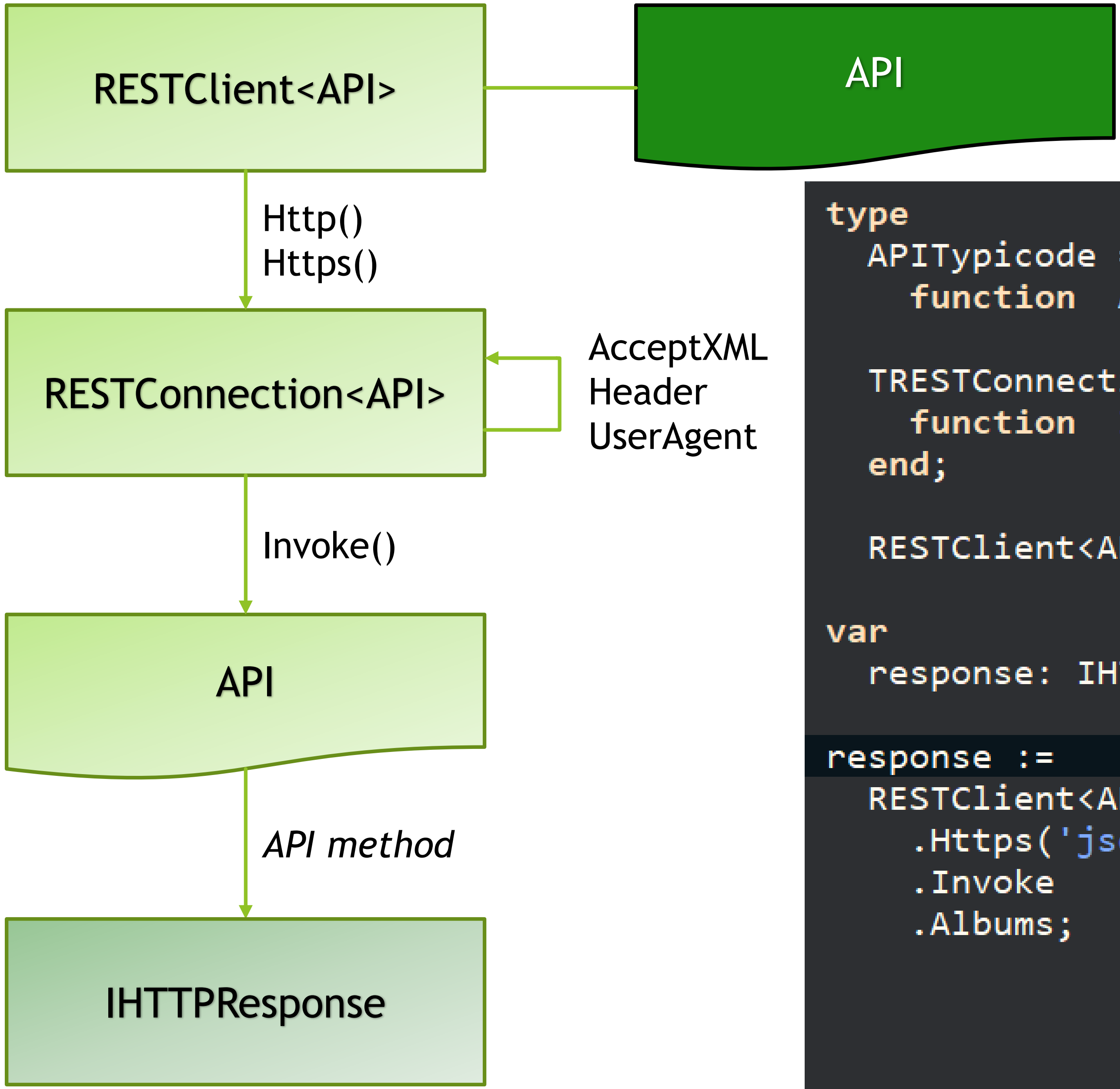
```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-

  TRESTConnection<API:IInterface> = record
    constructor Create(const uri: string);
    function  AcceptXML: TRESTConnection<API>; inline;
    function  Header(const name, value: string): TRESTConnec
    function  UserAgent(const agentName: string): TRESTConne
    function  Invoke: API;
    property Config: TRESTConnectionConfig read FConfig;
  end;

  RESTClient<API: IInterface> = record

RESTClient<APITypicode>
  .Https('jsonplaceholder.typicode.com', '/')
  .Invoke
  .
```



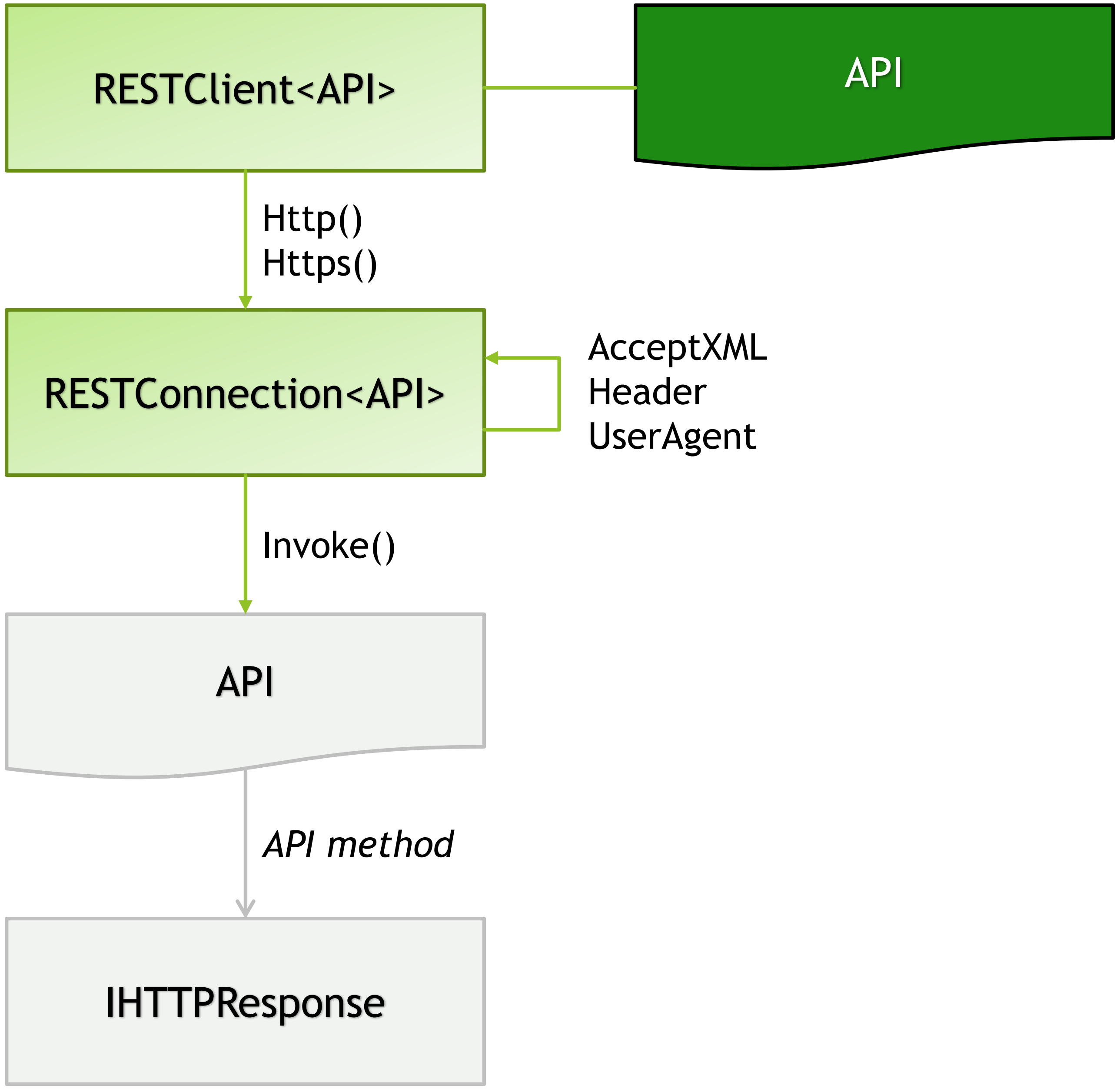


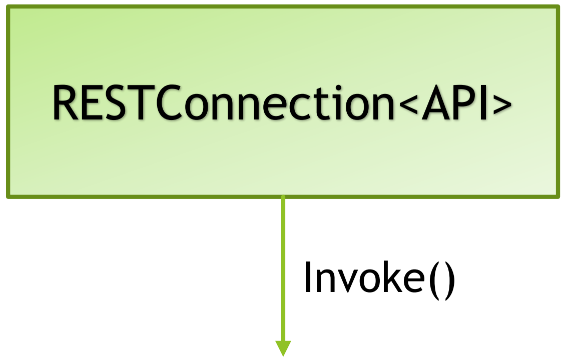
```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-
    function  Albums: IHTTPResponse;

  TRESTConnection<API:IInterface> = record
    constructor Create(const uri: string);
    function  AcceptXML: TRESTConnection<API>; inline;
    function  Header(const name, value: string): TRESTConnec
    function  UserAgent(const agentName: string): TRESTConne
    function  Invoke: API;
    property Config: TRESTConnectionConfig read FConfig;
  end;

  RESTClient<API: IInterface> = record

RESTClient<APITypicode>
  .Https('jsonplaceholder.typicode.com', '/')
  .Invoke
  .
```

```
function      Albums: IHTTPResponse;
function      Comments: IHTTPResponse;
function      QueryInterface(const IID: TGUID; out
function      _AddRef: Integer;
function      _Release: Integer;
```

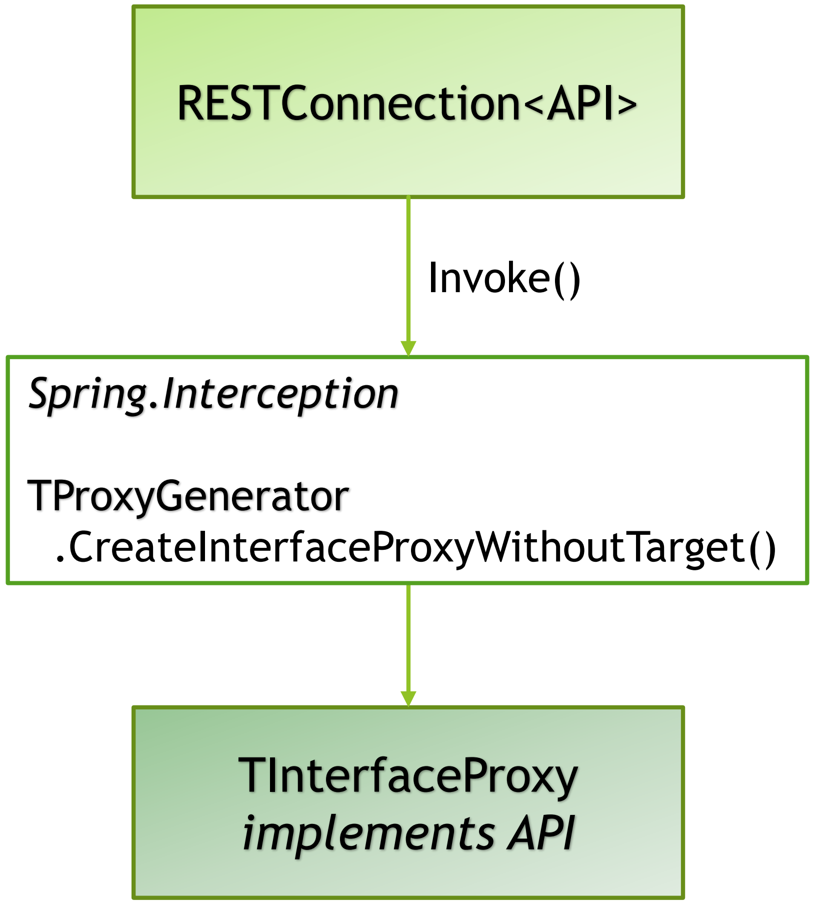


```
type
  APITypicode = interface(IInvokable) ['{54E1D5F5-EE4B-4F69-
    function  Albums: IHTTPResponse;

  TRESTConnection<API:IInterface> = record
    function  Invoke: API;
  end;

  RESTClient<API: IInterface> = record

var
  response: IHTTPResponse;

response :=
  RESTClient<APITypicode>
    .Https('jsonplaceholder.typicode.com', '/')
    .Invoke
    .Albums;
```

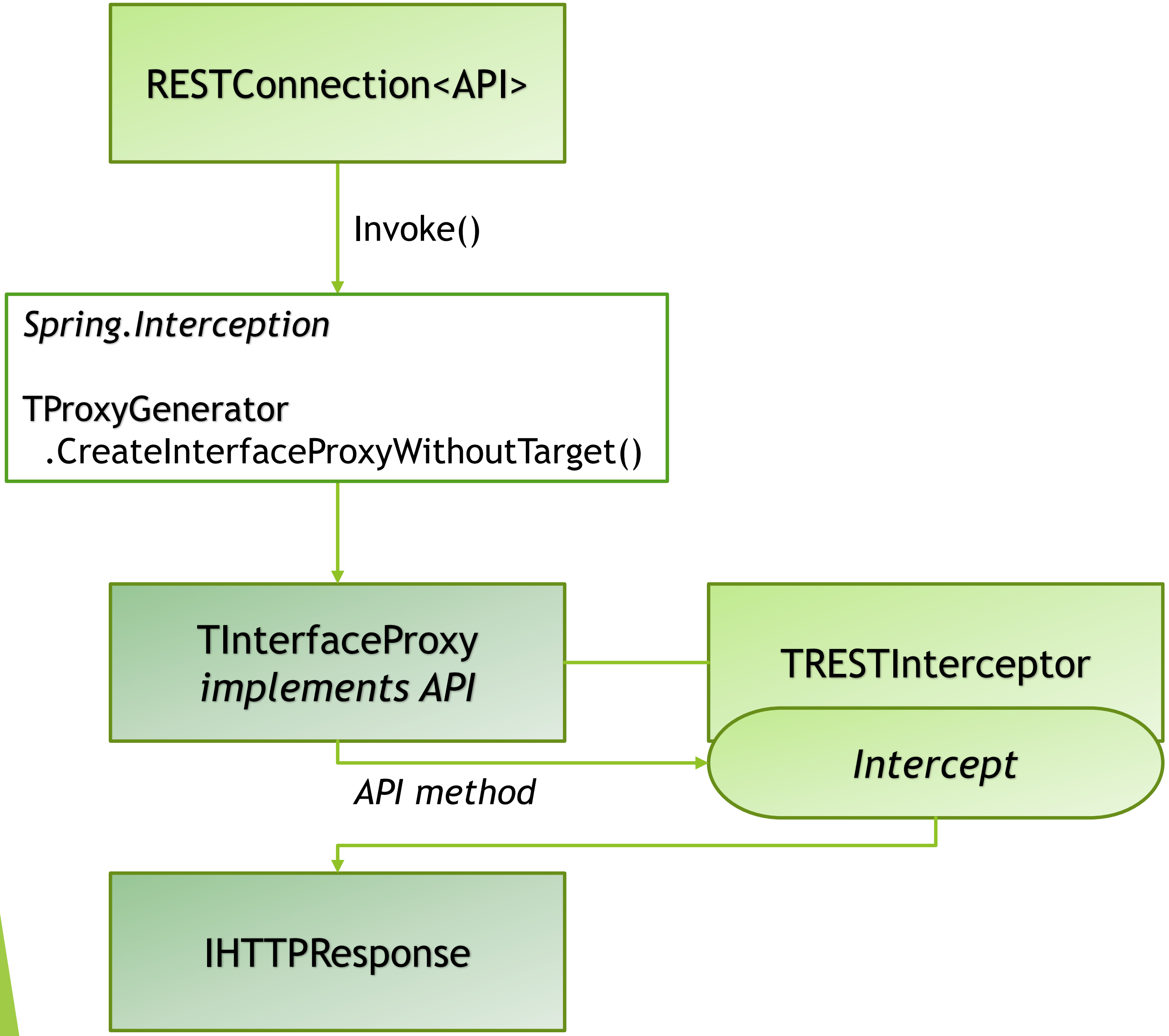

RESTClient<API>
API
Http()
Https()
RESTConnection<API>
AcceptXML
Header
UserAgent
Invoke()
API
API method
IHTTPResponse


RESTConnection<API>
Invoke()

Interception and dynamic proxy



```
type
  TRESTConnection<API:IInterface> = record
    function  Invoke: API;
  end;

function TRESTConnection<API>.Invoke: API;
begin
  Result := TProxyGenerator
    .CreateInterfaceProxyWithoutTarget<API>(
end;
```



```
type
  TRESTConnection<API:IInterface> = record
    function  Invoke: API;
  end;

  TRESTInterceptor = class(TInterfacedObject,
  public
    constructor Create(typInfo: PTypeInfo; co
    procedure Intercept(const invocation: IIn
  end;

function TRESTConnection<API>.Invoke: API;
begin
  Result := TProxyGenerator
    .CreateInterfaceProxyWithoutTarget<API>(
      TRESTInterceptor.Create(TypeInfo(API),
end;
```

“

THE END

”

european 4D conference